



UNIVERSIDADE DE SÃO PAULO

ENGENHARIA DE COMPUTAÇÃO

SSC0600 - INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO I

MATRIZES ESPARSAS

Autor:

Ivan Mateus DE LIMA
AZEVEDO

Professor:

Dr. Adenilso DA SILVA
SIMÃO

Nº USP:

10525602

15 de junho de 2018

1 Introdução

Este trabalho consiste em um programa que implementa uma matriz esparsa e suas operações básicas, como adicionar elementos nela, calcular a soma dos elementos das linhas e das colunas, apagar a matriz, etc. Além disso, também foi implementado utilizando as matrizes esparsas o método de solução de sistemas lineares de Gauss-Seidel.

2 Descrição do projeto

2.1 Ambiente de desenvolvimento

O código foi escrito na plataforma Linux através dos editores de texto Atom e Sublime Text e o projeto foi disponibilizado na plataforma de hospedagem de código-fonte GitHub.

2.2 Compilador utilizado

Foi utilizada a versão 5.4.0 do compilador GCC. E a plataforma de compilação da versão binária, i686.

2.3 Códigos-fonte

Para executar o programa, os seguintes arquivos são necessários e também é necessário ter a lib *math.h*, para poder utilizar a função *fabs* e também a lib *time.h*, para poder usar as funções *rand* e *srand*, instaladas na máquina:

1. main.c
2. cell.c
3. cell.h
4. matrix.c
5. matrix.h
6. gaussseidel.c
7. gaussseidel.h

3 Tutorial

3.1 Como compilar

Para compilar o programa é necessário ter o GCC instalado no micro utilizado. Após isso, abra o terminal, vá para o diretório em que os arquivos acima estão salvos e digite o seguinte comando:

```
$ gcc main.c cell.c matrix.c gaussseidel.c -o main -lm
```

3.2 Como executar

Caso queira executar o programa normalmente, basta digitar no terminal o comando abaixo. Ele abre o menu de opções.

```
$ ./main
```

Caso queira utilizar o programa para resolver um sistema linear, basta digitar:

```
$ ./main -gaussseidel
```

3.3 Entradas-exemplo

Ao executar o programa normalmente, o menu abaixo será exibido:

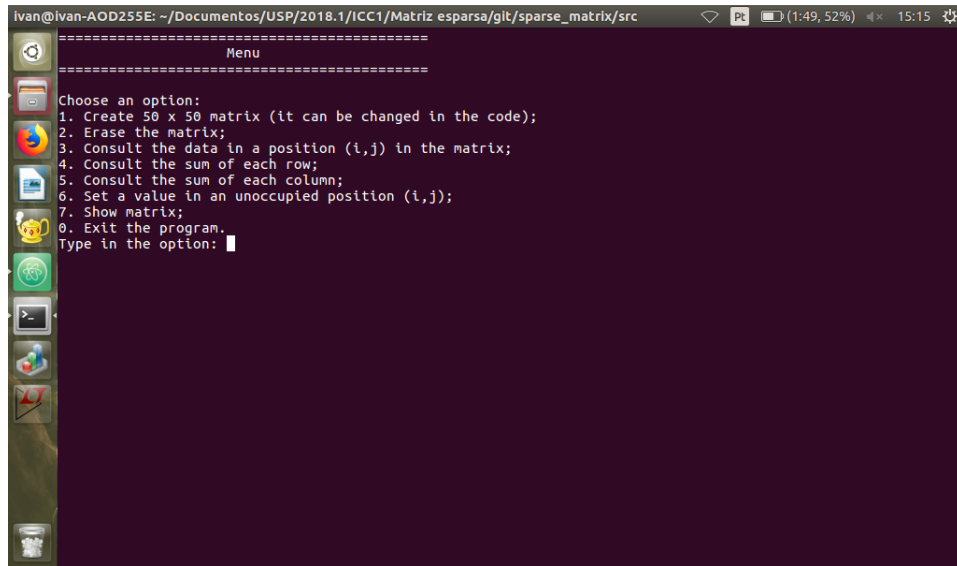


Figura 1: Figura que mostra o menu de opções.

Se o usuário escolher a opção 1, ele deverá informar quantos números não-nulos ele deseja adicionar na matriz.

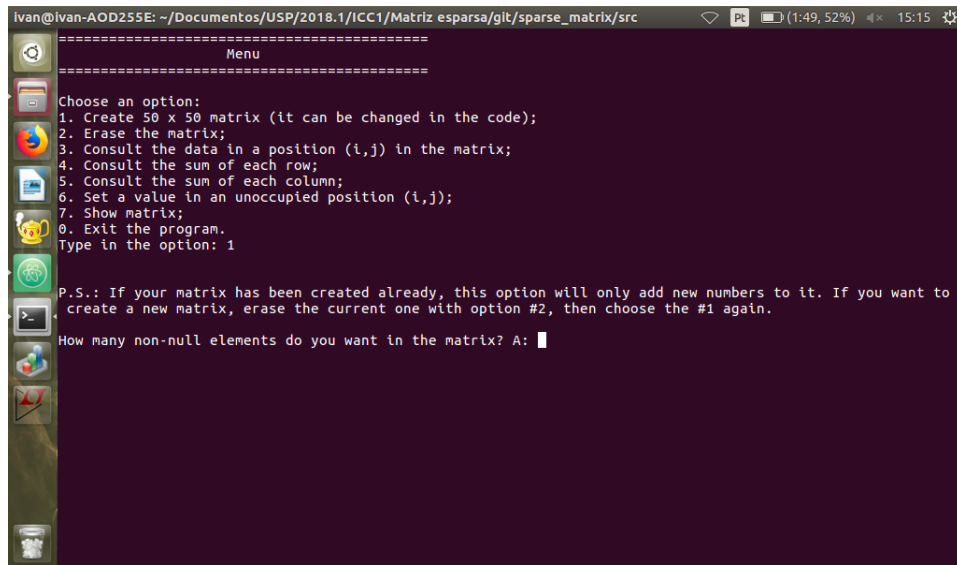


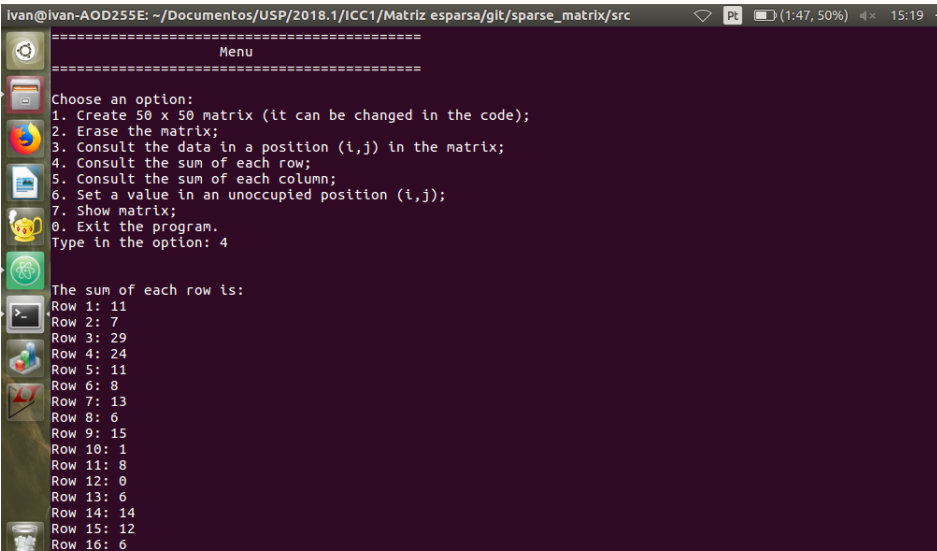
Figura 2: Figura que mostra o resultado da escolha da opção 1.

Caso escolha a opção 3, o usuário deverá escolher a posição que deseja consultar informando a linha e a coluna. Para facilitar, foi escolhida a posição (1,12), a qual é possível enxergar na figura.

[illegible]

Figura 5: Figura que mostra o resultado da escolha da opção 3 para a posição (1,12).

Caso escolha a opção 4, o programa exibirá o resultado da soma das linhas para cada linha da matriz.



```
Ivan@Ivan-AOD255E: ~/Documentos/USP/2018.1/ICC1/Matrix esparsa/glt/sparsa_matrix/src
=====
Menu
=====

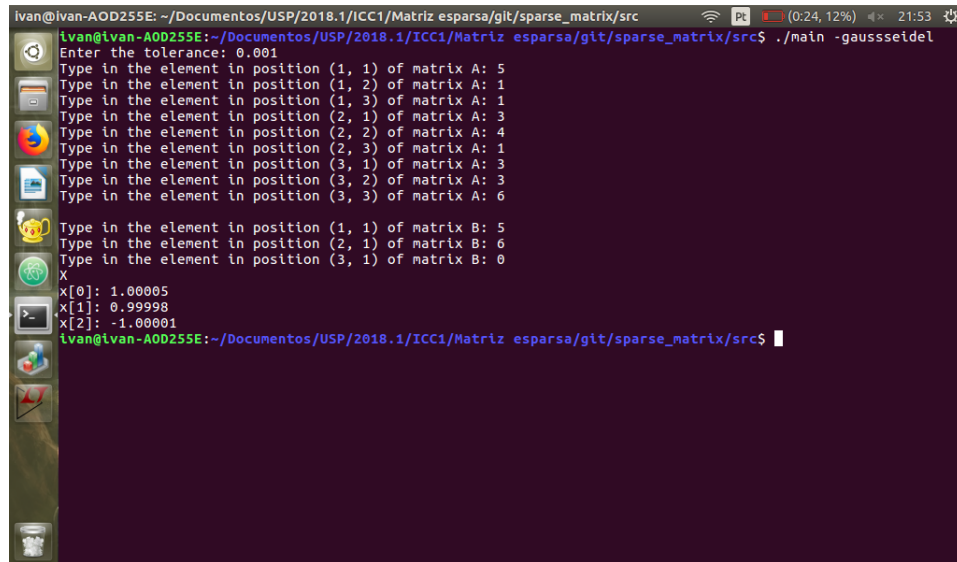
Choose an option:
1. Create 50 x 50 matrix (it can be changed in the code);
2. Erase the matrix;
3. Consult the data in a position (i,j) in the matrix;
4. Consult the sum of each row;
5. Consult the sum of each column;
6. Set a value in an unoccupied position (i,j);
7. Show matrix;
0. Exit the program.
Type in the option: 4

The sum of each row is:
Row 1: 11
Row 2: 7
Row 3: 29
Row 4: 24
Row 5: 11
Row 6: 8
Row 7: 13
Row 8: 6
Row 9: 15
Row 10: 1
Row 11: 8
Row 12: 0
Row 13: 6
Row 14: 14
Row 15: 12
Row 16: 6
```

Figura 6: Figura que mostra o resultado da escolha da opção 4.

Caso escolha a opção 5, o programa exibirá o resultado da soma das colunas para cada coluna da matriz.

Ao executar o programa, será pedida a tolerância desejada e os elementos das matrizes. Após isso, o método será aplicado até se chegar nessa tolerância. Por fim, o resultado será exibido.



```
ivan@ivan-AOD255E: ~/Documentos/USP/2018.1/ICC1/Matriz esparsa/git/sparse_matrix/src
ivan@ivan-AOD255E:~/Documentos/USP/2018.1/ICC1/Matriz esparsa/git/sparse_matrix/src$ ./main -gaussseidel
Enter the tolerance: 0.001
Type in the element in position (1, 1) of matrix A: 5
Type in the element in position (1, 2) of matrix A: 1
Type in the element in position (1, 3) of matrix A: 1
Type in the element in position (2, 1) of matrix A: 3
Type in the element in position (2, 2) of matrix A: 4
Type in the element in position (2, 3) of matrix A: 1
Type in the element in position (3, 1) of matrix A: 3
Type in the element in position (3, 2) of matrix A: 3
Type in the element in position (3, 3) of matrix A: 6
Type in the element in position (1, 1) of matrix B: 5
Type in the element in position (2, 1) of matrix B: 6
Type in the element in position (3, 1) of matrix B: 0
X
x[0]: 1.00005
x[1]: 0.99998
x[2]: -1.00001
ivan@ivan-AOD255E:~/Documentos/USP/2018.1/ICC1/Matriz esparsa/git/sparse_matrix/src$
```

Figura 11: Figura que exibe o resultado da operação.

4 Observações

É válido afirmar que o programa funciona muito bem para a criação de matrizes esparsas, visto que foram realizados testes com matrizes de até 1.000.000 x 1.000.000, com a inserção de 1.000.000 de elementos. Porém, a impressão de uma matriz dessa ordem é inviável, pois o processo se torna muito demorado. Portanto, caso se queira fazer testes desta magnitude, uma dica seria comentar no código a linha que imprime a matriz.

Outra coisa a ser dita é que, caso se queira definir as dimensões da matriz (tanto para as matrizes esparsas quanto para a solução de sistemas através do Gauss-Seidel), deve-se mudar o valor das constantes ROWS e COLS (linhas e colunas, respectivamente) no código, na área de definição de constantes do arquivo *matrix.h*. Por padrão, o valor foi deixado em 100 x 100. Além disso, os valores dos elementos não-nulos vão de uma faixa de 1 a 9, o que também pode ser mudado na mesma área (o nome da constante é MAXDATA).